

基于以太坊状态数据库的攻击与防御方案

高镇¹, 张东彬¹, 田潇²

(1. 天津大学电气自动化与信息工程学院, 天津 300072;
2. 南京慧链和信数字信息科技研究院, 江苏 南京 210012)

摘要: 以太坊是第二代区块链平台的典型代表, 其最大特点是能够通过智能合约来支持功能丰富的分布式应用。另外, 为了提高交易验证效率, 以太坊使用了本地数据库来存储账户状态, 并基于区块头内的状态树根保证状态数据的完整性。但是研究工作表明, 本地数据库存在被篡改的安全隐患, 且攻击者可以基于被篡改的账户状态发出非法交易, 从而牟取不正当利益。简要描述了这类针对本地状态数据库的安全漏洞, 并分析了攻击成功的前提条件; 在此基础上, 与工作量证明共识机制下两种常见的安全威胁进行了对比, 发现在攻击者控制相同挖矿算力的条件下, 基于状态数据库的攻击带来的安全风险更高, 攻击成功概率趋近于 100%。为应对存在的安全威胁, 提出了一套切实可行的攻击检测与防御方案, 并在以太坊源码的基础上加入了二次验证与数据恢复过程。通过单机多线程的实验测试评估了所提方案的可行性与复杂度。实验结果表明: 改进后的以太坊系统具备了针对状态数据库篡改的容错能力, 且该方案同样适用于超级账本等其他基于本地数据库进行交易验证的区块链平台。此外, 通过统计二次验证的时间和哈希计算次数, 证明了所提方案带来的时间与计算开销并不显著, 对现有系统的性能影响不大, 具有良好的适用性。

关键词: 以太坊; 状态数据库; 状态篡改; 非法交易; 攻击检测; 攻击防御; 篡改容错

中图分类号: TP 311.1

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2022013

Defense scheme for the world state based attack in Ethereum

GAO Zhen¹, ZHANG Dongbin¹, TIAN Xiao²

1. School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China
2. Nanjing Research Institute for Huilian Digital Information Technology, Nanjing 210012, China

Abstract: Ethereum is taken as the representative platform of the second generation of blockchain system. Ethereum can support development of different distributed applications by running smart contracts. Local database is used to store the account state (named world state) for efficient validation of transactions, and the state root is stored in the block header to guarantee the integrity of the state. However, some researches revealed that the local database could

收稿日期: 2021-10-08; 修回日期: 2022-02-15

通信作者: 高镇, zgao@tju.edu.cn

基金项目: 天津市自然科学基金 (19JCYBJC15700)

Foundation Item: The Natural Science Foundation of Tianjin (19JCYBJC15700)

引用格式: 高镇, 张东彬, 田潇. 基于以太坊状态数据库的攻击与防御方案[J]. 网络与信息安全学报, 2022, 8(2): 64-72.

Citation Format: GAO Z, ZHANG D B, TIAN X. Defense scheme for the world state based attack in Ethereum[J]. Chinese Journal of Network and Information Security, 2022, 8(2): 64-72.

be easily tempered with, and attackers can issue illegal transactions based on the modified account state to obtain illegitimate benefits. This world-state based security problem was introduced, and the preconditions for attack were analyzed. Compared with the two common security threats under the PoW (proof of work) consensus, it was found that when the attacker controls the same mining computing power, the world-state based attack brought higher risk, and the success rate approached 100%. In order to deal with this threat, a practical scheme for attack detection and defense was proposed accordingly. The secondary verification and data recovery process were added to the Ethereum source code. The feasibility and complexity of the proposed scheme was evaluated with single-machine multi-threading experiments. The proposed scheme improves Ethereum's tolerance to malicious tampering of account state, and is applicable to other blockchain platforms applying local database for transaction validation, such as Hyperledger Fabric. In addition, the time and computational overhead brought by the proposed scheme are not prominent, so it has good applicability and induces acceptable impact on the performance of original system.

Keywords: Ethereum, world state, state modification, invalid transactions, attack detection, attack defense, fault tolerance to state modification

0 引言

以太坊由 Vitalik Buterin 于 2013 年提出^[1], 被广泛认为是第二代区块链平台^[2]。相对于比特币系统^[3], 以太坊引入了账户和智能合约的概念^[2,4]。每个账户的最新状态被称为世界状态, 并被保存在本地数据库中。通过这种方式, 以太坊有效提高了查询和交易验证的效率。状态数据记录着许多有价值的信息, 如账户余额和交易计数等^[5]。而智能合约是分布在区块链网络中的代码, 可以被理解为一个计算机交易协议^[6]。在区块链提供的可信环境中, 智能合约在每个节点上独立执行, 保证执行结果的真实性和唯一性^[4,7]。

由于状态数据和智能合约的多样性, 以太坊已经广泛应用于许多领域。例如, Gems 是一个建立在以太坊基础上的分布式平台, 任何人都可以发布任务, 并采用在线支付报酬的方式来雇佣工人完成任务^[8]; 文献[9]利用 IPFS(inter planetary file system)的优势将文档存储在分散的文件系统上, 并利用智能合约来管理和规范创建者与开发者之间的文档版本; EdgeBoT 是一个基于概念证明(PoC, proof of concept)和智能合约的物联网平台, 可以使边缘设备直接与第三方直接交换数据, 而不需要中间商^[10]。

然而, 以太坊还面临一些安全问题。除了传统的自私挖矿(self-mining)^[11]和双重支付(double-spending)攻击^[12]之外, 文献[13]提出通

过篡改以太坊中账户的状态数据, 可以成功地发布一个无效交易。本文对这种攻击进行了详细的介绍, 并分析了攻击成功的条件和影响因素; 分析了对这种攻击的检测方法和数据恢复方案; 通过实验证明了所提方案的可行性, 通过时间开销评估了计算复杂度。

1 基于状态数据库的攻击

区块链的链上数据采用单向哈希函数连接在一起, 几乎不可能被篡改。但以太坊的状态数据存储于节点的本地数据库, 所以存在被篡改的风险。根据以太坊的共识机制, 各个矿工竞争挖矿, 这使得基于被篡改后的余额而发起的无效交易有随时被取消的可能。所以本节首先介绍了以太坊的共识算法和状态数据存储结构, 在此基础上介绍了基于状态数据库的攻击方式和成功条件。

1.1 PoW 共识

以太坊和比特币系统中使用的共识协议是工作量证明(PoW, proof of work)^[14]。当产生新的区块时, 网络中的矿工需要通过遍历来找到一个随机数以满足哈希值的范围要求, 这个过程通常被称为挖矿。成功的矿工会因付出的算力而得到报酬。由于挖矿是一系列哈希计算, 所以区块链矿工的计算能力通常被称为哈希率, 即每秒计算的哈希次数。挖矿速度由哈希率和挖矿难度决定^[15]。如果挖矿难度固定, 那么算力越高就意味着在较短时间内求解成功的概率越大。

由于网络的传播延迟,在某些时刻,区块链网络中可能存在具有相同长度的多条链(或称为分支)。当其中一条分支超过其他分支后,它将成为规范链,而其他分支中的交易将被退回给交易池,等待被后续区块打包。因此,在交易被区块链最终承认之前,通常需要等待一定数量的区块确认^[16]。当交易被一个区块打包之后,它将获得第一个确认,并且每个后续的区块都将为它添加一个新的确认。事实上,一个交易获得的确认数越多,那么该交易被撤销的概率就越小,但它的确认时间会越长。比特币和以太坊的确认区块数量(记为 N_c)分别为 6 个和 12 个^[17-18]。针对不同的场景,可以对 N_c 进行优化,从而使交易确认速度和系统安全性之间达成平衡^[19]。需要注意的是,区块链用户只能根据所依附节点的账本进行交易确认。因为共识机制保证了不同节点上账本的一致性,所以这一过程通常是可靠的。

1.2 状态树与状态验证

以太坊本质上是一个基于交易的有限状态

机。每个账户的状态数据包含当前的余额和交易计数器,新的状态(σ_{N+1})可以根据其父状态(σ_N)和交易(T)经过转换后获得^[5]。如果将以以太坊状态转换函数表示为 γ ,那么转换关系就可以表示为

$$\sigma_{N+1} = \gamma(\sigma_N, T) \quad (1)$$

如图 1 所示,以太坊中所有账户的状态数据存储于 MPT (Merkle patricia tree) 的最底层节点。状态树的根哈希值(即状态树根(stateRoot))被存储于区块头中,其计算过程为:通过递归自下而上对状态树中的节点逐级进行哈希运算,并且每个节点以哈希值(hashNode)的形式保存于上一级节点中。为实现持久化存储,每个节点的哈希值和节点数据以 key-value 的形式存储在数据库中。而在查询或更改某个账户的状态时,需要沿着账户地址,根据上一级节点存储的 hashNode 从数据库中加载下一级节点的数据,直到找到保存有余额的最底层节点。

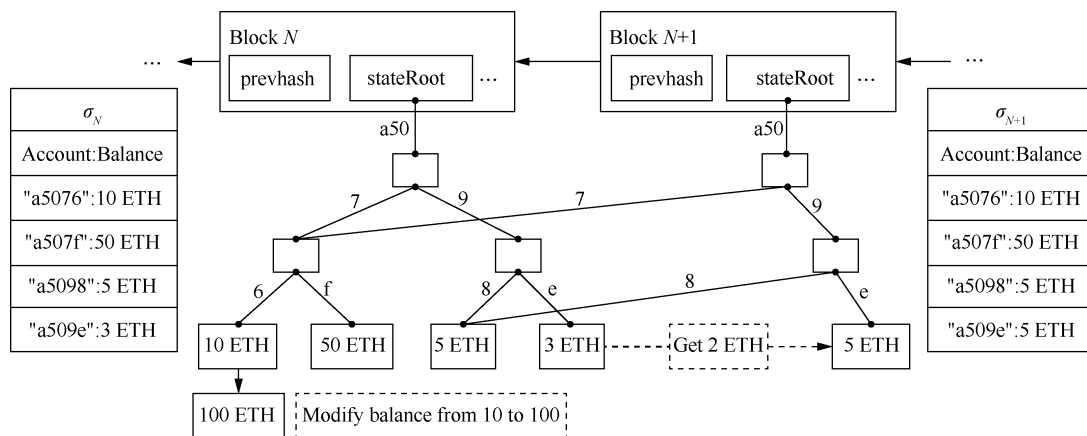


图 1 以太坊状态树架构(以 4 个账户为例)

Figure 1 Ethereum state tree structure (taking 4 accounts as an example)

区块链节点收到新区块后验证其 stateRoot 的过程可以分为以下几步。① 加载本地存储的状态树以获取并检查转出账户的余额(该过程也会加载转入账户所在的状态树路径,而除了转入与转出账户所在的路径,其他路径上存储的状态数据不会发生改变,那么其中的 hashNode 也不变,所以不需要加载)。② 按照式(1)更新交易后的状态。③ 根据新的状态计算更新后的状态树根,并与区块头中的 stateRoot 进行比较。当两个哈希树根匹配时,节点才接受该区块;否则,该区块将

被视为无效。例如,图 1 中的账户“a509e”从一个交易中获得两个 ETH,那么其余额会从 3 个 ETH 变为 5 个 ETH,并且新的状态树根将沿着路径“e—9—a50”重新计算出来;但其他路径,如“6—7—a50”中的子路径“6—7”就不需要被加载到内存中,以 hashNode 的形式参与“a50”处的运算即可。

1.3 传统的安全问题

PoW 共识下的公有区块链有两个安全威胁,分别是自私挖矿和双重支付。而它们的相似之处

是攻击者利用最长链原则来牟取私人利益。在自私挖矿攻击中，一个算力强大的矿工或由多个矿工组成的采矿池故意囤积一个或多个新生成的区块，而不把它们广播给网络中的其他节点。然后，在适当的时候再把包含囤积区块的秘密链通知给其他节点，以取代诚实矿工的有效区块。文献[11]证明：即使攻击者不控制大部分的算力，也可以比相同条件下的诚实挖矿获取到更多的回报。文献[20]的分析表明以太坊比特币更容易受到自私挖矿攻击。双重支付与之类似：首先，攻击者在交易确认之后，花费代币并获得相应的商品或服务；然后，它通过发布一个秘密且更长的链迫使网络中的节点撤销原有的交易^[21]，于是，攻击者花费的代币便会被退回，并可以用于二次消费。在上述情况下，攻击者控制的算力占比越大，则攻击成功的概率将越高。一旦攻击者的算力占比超过整个区块链网络的50%，那么攻击最终一定会成功，因此，称之为51%攻击^[22]。

1.4 基于账户状态的攻击

由于状态数据存储在每个节点的本地数据库中（如LevelDB），因此攻击者可以通过数据库接口函数修改其数值。假设在攻击过程中，攻击者通过数据库接口增加了某些账户的余额，并尝试与其他账户进行交易，交易金额远大于账户的实际余额但小于修改后的余额。接下来，如文献[13]所述，被篡改状态数据的账户可以成功地发出无效的交易，而不会出现任何警告或错误，并且所有被篡改数据的节点都会接受该交易。出现此问题的原因是以太坊不检查用于链上交易的状态数据的正确性，也不将本地状态与其他节点的状态进行比较。利用此漏洞，攻击者可以任意增加余额，无节制地花费实际上并不存在的代币。Hyperledger Fabric是另一个使用本地状态数据库进行高效交易验证的区块链平台，文献[23]证明Fabric也无法避免类似的攻击。但以上两篇文献都只讨论了这种基于账户状态的攻击的可能性，而没有分析交易被最终确认的条件和恢复正常数据的方法。

1.5 攻击成功的条件

尽管无效交易能够被成功地发布和验证，但它们可能仍无法被最终确认。由于转出金额大于

实际余额，诚实节点并不会接受该交易。所以只有被篡改的节点才能将这笔无效交易打包到新的区块中，然后广播给网络中的其他节点。随后，其他被篡改的节点将成功接受该区块，并在该区块的基础上继续产生下一个区块。但是当诚实节点收到该区块后，由于被篡改账户的余额不足以支付交易，所以诚实节点将丢弃该区块，更不会接受其子区块。如图2所示，这时网络中会出现两条分支，一条由被篡改的节点发布（记为chainA），另一条链由诚实节点发布（记为chainB）。根据最长链原则，当chainB的长度超过chainA时，被篡改的节点将舍弃chainA并同步chainB。但是，即使chainA的长度大于chainB，由于转出账户的余额不足，诚实节点仍无法同步chainA。

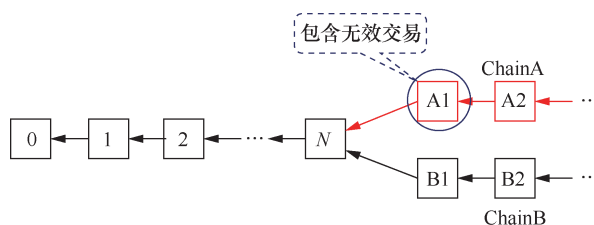


图2 在区块高度为 N 处遭到攻击后，区块链网络出现两条分支示意
Figure 2 Two branches on tampered nodes with attack at block height N

只有当无效交易获得了足够多的区块确认之后，攻击者才能真正地把代币花费出去，攻击才能够成功。所以攻击成功的首要条件就是被篡改的节点产生这笔交易的确认信息。在此基础上，该交易能否被确认取决于在确认之前chainA的区块高度能否一直大于或等于chainB的高度。如果chainA新增的区块数达到了所需要的确认数量(N_c)，并且这个过程中chainA没有被chainB取代，那么第 $N+1$ 个区块（即图2中的A1区块）中包含的无效交易将被确认。

此外，攻击能否成功还取决于诚实节点的后续区块（区块高度大于 N ）是否包含涉及被篡改账户的交易。假设该笔有效交易由正常节点打包在区块 $N+n$ 中，其中 $1 < n < N_c$ ，那么该区块将被所有正常节点接受。但对于被篡改的节点而言，基于篡改后的数据计算出的哈希树根与该区块头中的哈希树根不匹配，所以该区块将被拒绝。在这种情况下，即使包含该区块的chainB长度比

chainA 大,被篡改节点也不会接受。这使得 chainA 与 chainB 无法相互同步。换句话说,在无效交易被打包到被篡改节点的区块之后,攻击者可以通过诚实节点再发起一笔有效交易(用于攻击的辅助交易)。并且,攻击者可以为辅助交易设置较高的交易费用,以便诚实节点优先对该交易进行打包。如果辅助交易被一个诚实节点打包到一个高度小于 $N+N_c$ 的区块中,那么这个区块在所有被篡改的节点上都会被判定为无效。因此,chainA 的长度一定可以达到 N_c ,并且以 100% 的概率确认无效交易。

另外,如果 chainA 在高度达到 $N+N_c$ 之前没有被 chainB 替换掉,那么即使攻击者没有发起辅助交易,无效交易仍然可以被确认。这可以理解为一组被篡改的节点和正常节点之间的竞争,单次攻击的成功概率取决于两组节点的计算能力和所需要的区块确认数。被篡改节点的算力占比越高或者区块确认数越少,那么单次攻击的成功概率就越高。此外,每次攻击失败后,无效的交易都将返回到交易池中。由于在没有人干预的情况下,被篡改的节点无法检测到攻击并恢复被篡改的数据,所以这些节点会重新打包无效的交易,从而再次发起攻击。基于状态数据库的攻击对于攻击发起者而言代价并不大,所以这种攻击可以自动地无限进行下去,直到最终成功。当攻击的次数足够多时,理论上的成功概率将无限趋近于 100%。

基于状态数据库的攻击与传统的自私挖矿和双重支付在攻击方式上都需要依赖足够的算力。如果控制的节点算力占比较小,那么自私挖矿和双重支付攻击成功的概率较低。根据文献[12],当控制的节点算力占比为 10% 并且系统要求的确认区块数量为 6 个时,双重支付攻击成功的概率仅为 0.059%。而根据文献[11],当控制的算力占比大于 33% 时,自私挖矿才是严格有利的;并且自私挖矿主要影响挖矿收益,其安全威胁相对双重支付攻击更小。但在基于状态数据库的攻击中,攻击者可以采取辅助交易或者使攻击无限次发起的手段,来使攻击成功,即花费比正常余额更多的代币。所以即使在控制算力较小的情况下,基于状态数据库的攻击也会造成极高的安全风险。

2 检测与恢复方法

本节首先分析了检测本地状态数据的具体方法。而后通过具体的方案证明了:即使状态数据库被篡改,节点也可以根据本地历史状态数据和交易恢复出正确的数据。

2.1 检测方法与流程

以太坊系统没有检查历史状态数据的正确性,才导致基于状态数据库的攻击能够成功。因此,解决这个问题的最好方法就是让节点主动检查本地的余额是否正确。如图 3 所示,在区块高度为 N 时,账户“a5076”的余额遭到篡改,并且随后的第 $N+1$ 个区块中包含基于篡改后余额的交易。被篡改的节点接收到其他矿工传来的第 $N+1$ 个区块时,直接检查新区块中的交易和状态树根是否符合要求,而忽略了第 N 个区块状态数据的正确性,所以无法检测出本地的余额已经被篡改。

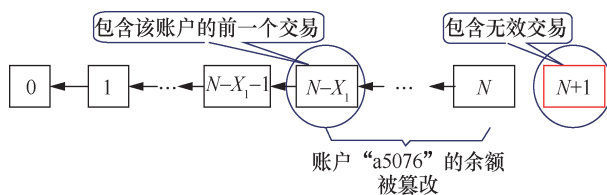


图 3 包含被篡改账户的区块链示意
Figure 3 A blockchain structure diagram containing a tampered account

因此,一种解决方法是在验证第 $N+1$ 个区块之前,让节点根据世界状态重新计算当前的状态树根,并与存于链上的 stateRoot 进行比较,称这一过程为二次验证。但以太坊网络中的账户众多,完整地计算整个状态树几乎不可能。事实上,参考某一节点收到并首次验证第 N 个区块中状态树根的过程,因此另一种方法是只需要计算必要的账户所在的状态树路径即可,不必对整个状态树展开计算(未展开的状态树路径以 hashNode 的形式参与计算)。而二者的区别在于:在首次验证第 N 个区块时,节点只计算了第 N 个区块中交易涉及的账户在状态树中的路径;而在二次验证时,节点需要重新计算第 $N+1$ 个区块中交易涉及的账户在区块高度为 N 时的状态树路径。检测方案流程如图 4 所示。

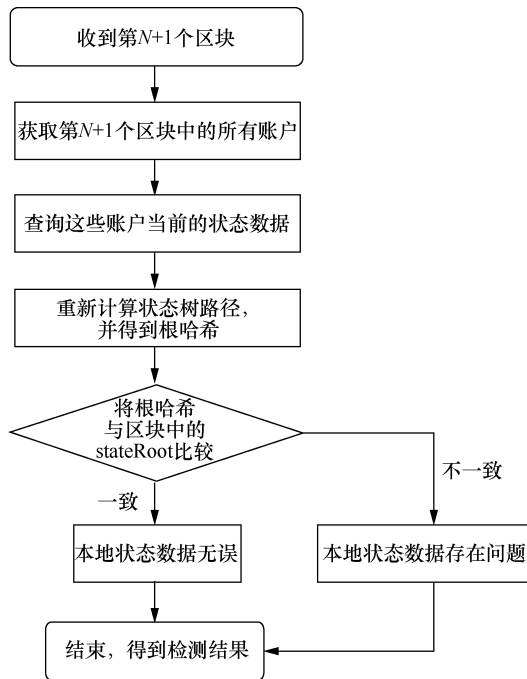


图4 检测方案流程
Figure 4 Detection process flow chart

值得注意的是, 上述两种计算过程都从状态树的最底层算起, 虽然计算路径可能存在差异, 但正常情况下, 最终结果应为同一个状态树根。而如果此时恶意攻击者发起了基于状态数据库的攻击, 那么第 $N+1$ 个区块中将包含恶意交易 (即关于被篡改账户的交易), 所以被篡改的账户所在的状态树路径也一定在二次验证所校验的路径范围之内。由于余额被篡改, 所以节点二次计算出的状态树根与链上的状态树根必然不同, 以此就可以判断出本地的余额遭到篡改。即使节点被攻击, 二次验证也会按照正常步骤执行, 检测结果并不会受到影响, 所以节点可以有效发现潜在的问题。

此外, 检测过程须不间断地进行, 即在每一个区块上链之前, 节点都需要二次验证前一个区块的状态数根。如果忽略了某个区块, 而其恰好包含无效的交易, 并且本地节点的余额遭到了篡改, 那么无效的交易和哈希树根就将写入本地区块链。随后即使该节点二次验证状态数根, 其根据也是错误的余额和状态树根, 所以就无法通过对比发现可能存在的问题。

2.2 恢复方法与流程

如果该节点能够保证二次验证每一个区块的

状态数根, 那么其链上的数据一定是正确的。而该节点一旦发现二次验证得到的状态树根与链上状态树根不一致, 就说明当前的状态数据遭到篡改。为了恢复正常的余额, 节点可以依据本地数据库中存储的历史状态数据和区块中的交易计算出当前区块高度下正确的状态数据。

因为难以精准地找到状态数据被篡改的账户, 所以需要遍历区块 $N+1$ 中的全部账户, 并重新计算它们当前的余额。图5以账户“a5076”为例, 描述了恢复状态数据的具体方法。首先, 节点需要找到该账户上一笔交易所在的区块。根据图3所示的包含账户“a5076”的区块链示意, 该节点需要回滚 X_1 个区块, 即上一笔交易所在的区块高度为 $N-X_1$ 。回滚 X_1 个区块是因为如果 N 处该账户的状态数据被篡改了, 那么不管在 $N-X_1$ 到 N 之间有多少个区块, 这些区块高度下查询到的该账户的余额都是被篡改后的。这是由MPT树的结构决定的: 在若干个区块之间, 如果一个账户没有发生交易, 那么这期间它的状态树路径不变, 并且对应相同的状态数据。所以篡改该账户的状态数据后, 只有通过区块高度为 $N-X_1-1$ 下的账户状态数据, 并按照式(1)重新执行区块 $N-X_1$ 中的交易, 才能计算出该账户正确的数据。

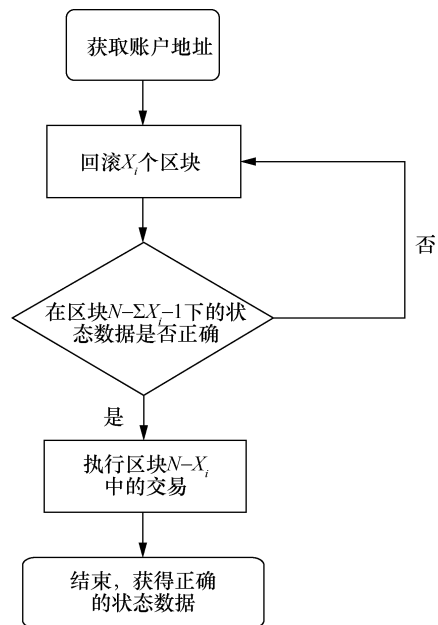


图5 单个账户状态数据恢复流程
Figure 5 Single account state-data recovery flow chart

但在正式恢复数据之前,节点还需要按照 2.1 节的方法二次验证区块 $N-X_1-1$ 下的状态数据是否正确。如果存在问题,那么节点需要继续向前寻找涉及该账户的交易和旧的状态数据(设回滚的区块数量为 X_2),并再次验证;如果数据无误,那么按照式(1)就可以计算出账户“a5076”在 $N-X_1$ 个区块之后正确的余额。

3 方案功能验证与复杂度评估

虽然节点通过检测能够及时发现本地状态数据遭到篡改,但前提条件是节点必须花费多余的算力和时间来对每一个区块进行二次验证。所以实验部分除了分析所提方案的有效性之外,也对算法的时间开销进行了评估。

本文实验在以太坊的 Golang 版本(geeth-windows-amd64-1.9.10)的基础上增加了二次验证和状态数据恢复的功能。实验的初始化配置如表 1 所示。本文搭建了一个具有 3 个节点的小型以太坊网络,作为进程运行的每个节点都配有不同的端口号,并直接与其他两个节点连接。其中一个节点为诚实节点,而另外两个节点的状态数据被篡改。

表 1 实验初始化配置
Table 1 Experiment initialization configuration

类型	参数
CPU	Intel(R) Core(TM) i5-11400 CPU
RAM	16.00 GB
运行系统	Microsoft Windows 10 Home Edition 64-bit
节点个数	3

首先,本实验验证了所提出的状态数据篡改检测方案。在检测开始之前,其中一个被篡改的节点发起若干交易并将它们打包到区块 $N+1$ 中。随后,按照程序设计,当另外两个节点接收到新区块后,会二次验证状态数据。如图 6 所示,另外一个被篡改的节点在二次验证后,发现当前的状态数据计算出的状态树根与区块 N 中存储的状态树根不一致。此时该节点会按照 2.2 节中的介绍恢复出正确的状态数据,然后再执行区块 $N+1$ 中的交易。

其次,在 200 次独立实验的基础上通过时间统计验证了本文方案的复杂度。如图 7 所示,当

系统中总的账户数量为 5 000 个时,分别统计了二次验证需要的哈希计算次数与验证平均所用的计算时间。例如,当区块 $N+1$ 涉及的账户个数为 200 时,哈希计算次数为 524 次,验证所用的平均时间为 2.3 ms。而经过测试,此时系统对整个新区块的验证时间(包含状态数根、难度值和余额等数据的验证)约为 26 ms,可见二次验证带来的时间开销并不显著。此外,通过观察可以得出结论:哈希计算次数以及二次验证计算时间与区块交易涉及的账户个数大致呈线性关系。如图 8 所示,本文固定了交易涉及的账户数量(100 个),然后在不同的账户数量下,测试了二次验证需要的时间。结果表明计算时间和哈希计算次数与状态树规模也呈线性关系。

```
header.stateRoot:[79 107 64 205 154 215 114 156 198 205 154 125 193
134 123 26 69 71 76 81 158 34 9 43 211 115 168 204 77 6 47 40]
> Parent-stateRoot:[65 37 246 46 160 84 136 24 247 182 142 18 164 6
9 184 129 73 219 209 249 122 126 180 1 246 157 27 132 216 79 190 13
7]
Recalculated stateRoot:[114 244 40 218 242 45 45 139 199 57 18 188
100 3 212 123 113 91 137 239 21 28 207 244 40 205 17 8 105 9 117 14
4]
StateRoot mismatch!
time of calculating : 1993700 ns
```

图 6 被篡改节点发现错误
Figure 6 A tampered node found error

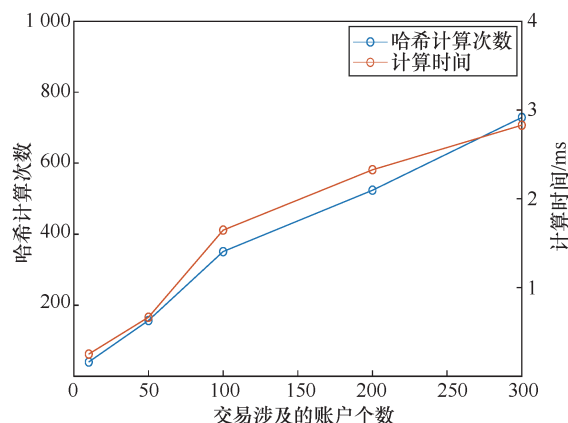


图 7 哈希计算次数/二次验证计算时间与区块交易涉及账户个数的关系
Figure 7 The relationship between the number of hash calculations/secondary verification calculation time and the number of accounts involved in block's transactions

4 结束语

以太坊通过引入本地状态数据库来提高交易验证效率,并通过存于区块头中的状态树根来保持状态数据的完整性。但是本地数据库可以通过数据库接口进行修改,且目前的以太坊并不会对本地状态数据库的完整性进行验证,

这使得基于篡改数据库来发起非法交易成为可能,并且理论上攻击成功的概率为100%。针对这类安全漏洞,本文提出一种可行的状态数据篡改检测方案以及相应的修复方案。基于单机多线程实验测试,验证了状态数据篡改检测方案和恢复方案的可行性,并对检测复杂度进行了评估。实验结果表明,虽然本文提出的检测方案会造成一定的算力与时间开销,却可以有效避免本地状态数据被篡改,并足以抵御攻击。其中检测时间与每个区块中交易所涉及的账户个数成正比。在具有5 000个账户的以太坊系统中,如果区块中的交易涉及200个账户,则二次验证时间约占整体区块上链时间开销的9%,可见代价较小。此外,本文所提出的方案同样适用于其他基于本地状态数据库进行交易验证的区块链系统。

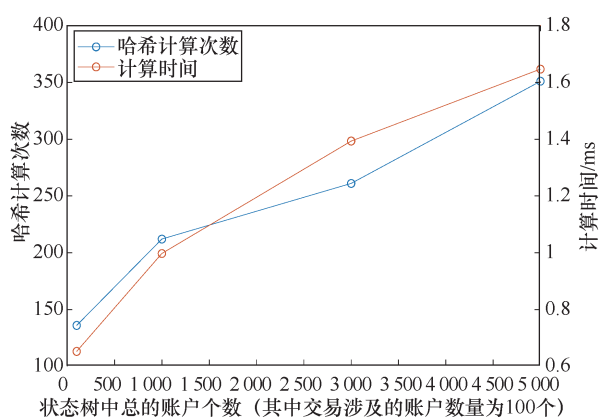


图8 哈希计算次数/二次验证计算时间与状态树规模的关系
Figure 8 The relationship between the number of hash calculations/secondary verification calculation time and the size of the state tree

参考文献:

- [1] BUTERIN V. Ethereum white paper[J]. GitHub repository, 2013, 1: 22-23.
- [2] KEHRLI J. Blockchain 2.0-from bitcoin transactions to smart contract applications[R]. 2016.
- [3] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system[R]. Manubot, 2019.
- [4] CHEN T, LI Z H, ZHU Y X, et al. Understanding Ethereum via graph analysis[J]. ACM Transactions on Internet Technology, 2020, 20(2): 1-32.
- [5] WOOD G. Ethereum: a secure decentralized generalized transaction ledger[J]. Ethereum Project Yellow Paper, 2014, 151(2014): 1-32.
- [6] BASHIR I. Mastering blockchain[M]. Packt Publishing Ltd, 2017.
- [7] WANG S, OUYANG L W, YUAN Y, et al. Blockchain-enabled smart contracts: architecture, applications, and future trends[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(11): 2266-2277.
- [8] K. O. R. O'reilly, gems protocol[EB].
- [9] NIZAMUDDIN N, SALAH K, AZAD M A, et al. Decentralized document version control using Ethereum blockchain and IPFS[J]. Computers & Electrical Engineering, 2019, 76: 183-197.
- [10] NAWAZ A, PEÑA QUERALTA J, GUAN J X, et al. Edge computing to secure IoT data ownership and trade with the Ethereum blockchain[J]. Sensors, 2020, 20(14): 3965.
- [11] EYAL I, SIRER E G. Majority is not enough: bitcoin mining is vulnerable[M]//Financial Cryptography and Data Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014: 436-454.
- [12] ROSENFELD M. Analysis of hashrate-based double spending[J]. arXiv preprint arXiv:1402.2009, 2014.
- [13] GAO Z, ZHANG D B, ZHANG J Z. A security problem in small scale private Ethereum network[C]//Proceedings of International Conference on Blockchain and Trustworthy Systems. 2020: 231-242.
- [14] ACH L M, MIHALJEVIC B, ZAGAR M. Comparative analysis of blockchain consensus algorithms[C]//Proceedings of 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2018: 1545-1550.
- [15] BISSIAS G, THIBODEAU D, LEVINE B N. Bonded mining: Difficulty adjustment by miner commitment[M]//Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, Cham, 2019: 372-390.
- [16] BONNEAU J, MILLER A, CLARK J, et al. SoK: research perspectives and challenges for bitcoin and cryptocurrencies[C]//Proceedings of 2015 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2015: 104-121.
- [17] WEBER I, GRAMOLI V, PONOMAREV A, et al. On availability for blockchain-based systems[C]//Proceedings of 2017 IEEE 36th Symposium on Reliable Distributed Systems. 2017: 64-73.
- [18] EZ M, FORNARI G, VARDANEGA T. The scalability challenge of Ethereum: an initial quantitative analysis[C]//Proceedings of 2019 IEEE International Conference on Service-Oriented System Engineering. 2019: 167-176.
- [19] RADBURY D. The problem with bitcoin[J]. Computer Fraud & Security, 2013, 2013(11): 5-8.
- [20] ENG C, NIU J Y. Selfish mining in ethereum[C]//Proceedings of 2019 IEEE 39th International Conference on Distributed Computing Systems. 2019: 1306-1316.
- [21] ANG J, LEE H N. Profitable double-spending attacks[J]. Applied Sciences, 2020, 10(23): 8477.

- [22] AYEED S, MARCO-GISBERT H. Assessing blockchain consensus and security mechanisms against the 51% attack[J]. Applied Sciences, 2019, 9(9): 1788.
- [23] GAO Z, ZHANG D B, ZHANG J Z. A security problem caused by the state database in Hyperledger Fabric[C]//Proceedings of International Conference on Frontiers in Cyber Security. 2020: 361-372.



张东彬(1998-), 男, 河南平顶山人, 天津大学硕士生, 主要研究方向为区块链平台的安全分析。

[作者简介]



高镇(1982-), 男, 河北张家口人, 天津大学副教授, 主要研究方向为容错信号处理与区块链技术与应用。



田潇(1992-), 男, 江苏南京人, 南京慧链和信数字信息科技研究院工程师, 主要研究方向为区块链平台的共识算法与隐私计算。